Weighted Sums as Filters, Memories, and Interference Systems

Introduction

The weighted sum is usually introduced as a simple arithmetic operation:

$$y = \sum_i w_i x_i$$

But this hides what the operation is really doing.

A weighted sum is simultaneously:

- a filter,
- a similarity detector,
- an interference mechanism,
- a compressed memory,
- and a projection operator.

Large parts of signal processing, neural networks, statistics, and harmonic analysis are built from systems of weighted sums, yet these connections are often presented separately.

This note takes a more intuitive and practical view.

---

1. The Ordinary Sum is Already a Filter

Consider the simple sum:

$s = x_1 + x_2 + \ldots + x_n$

This looks trivial, but it is already a weighted sum:

s = <x,1>

where:

1 = (1,1,…,1)

This means:

- the ordinary sum is a projection,
- specifically onto the constant vector.

That makes it a filter.

Cancellation and Energy Loss

Suppose:

x = (1,-1,1,-1)

Then:

$$\sum_i x_i = 0$$

Even though the vector contains substantial energy.

The sum completely annihilates the signal.

The squared energy:

$$\sum_i x_i^2$$

is nonzero, but the projection onto the (1,1,...,1) direction vanishes.

This is destructive interference.

The weighted sum therefore:

- amplifies some structures,
- suppresses others,
- and completely cancels patterns orthogonal to its filter direction.

This is exactly what a matched filter does.

---

2. Weighted Sums as Matched Filters

A weighted sum:

$$y = \langle w, x \rangle$$

detects how much the input resembles the weight pattern.

If:

- the input aligns with the weight vector,
- contributions add coherently,
- the output grows.

If:

- the input opposes the weight vector,
- cancellation occurs,
- the output shrinks or reverses sign.

The weight vector therefore defines:

- what patterns are amplified,
- what patterns are suppressed,
- and what patterns are annihilated.

This is the core operation behind:

- correlation,
- template matching,
- signal detection,

- and neuron activations.

---

3. The Weighted Sum as Associative Memory

Now consider storing associations:

$$x^\mu \to y^\mu$$

where:

- x^μ is an input vector,
- y^μ is the desired scalar output.

The weighted sum stores these associations in compressed form inside the weight vector.

Superposition Storage

One simple storage rule is:

$$w = \sum_\mu y^\mu x^\mu$$

Each example contributes:

- a directional preference,
- a projection tendency,
- a piece of memory.

The weight vector becomes a superposition of all previous examples.

Retrieval

Recall occurs through projection:

$$\hat{y} = \langle w, x \rangle$$

Expanding:

$$\hat{y} = \sum_{\mu} y^{\mu} \langle x^{\mu}, x \rangle$$

Similar stored vectors reinforce the response.
Dissimilar vectors mostly cancel.

Memory retrieval is therefore an interference process.

---

4. Under-Capacity Behavior

Suppose:

- vector dimension is large,
- stored examples are relatively few.

Then most vectors are nearly orthogonal.

The stored memories interfere only weakly.

SGD Under Capacity

With SGD:

- the weight vector gradually moves toward a configuration that satisfies all examples,
- once all examples are fitted,
- the gradient largely disappears,
- SGD stops moving.

The result is often:

- good fit,
- stable retrieval,
- large margins,
- and strong error correction.

However SGD does not necessarily find the best possible solution.

It finds:

- a reachable local trajectory,
- shaped by learning order,
- learning rate,
- initialization,
- and optimization noise.

Moore–Penrose Under Capacity

The Moore–Penrose solution instead computes:

$$w = X^{+}y$$

This gives:

- the minimum norm exact solution,
- globally balancing all examples simultaneously.

Compared to SGD:

- crosstalk is often lower,
- geometry is cleaner,
- interference is more balanced,
- and unused directions are minimized.

Under capacity, Moore–Penrose can therefore outperform SGD.

---

5. Error Correction Under Capacity

In the under-capacity regime:

- many stored projections cooperate,
- many directions support the correct answer,
- retrieval margins are large.

This creates natural error correction.

Small perturbations:

- partially cancel,
- average away,
- or remain orthogonal to dominant memory directions.

This is why distributed memories degrade gracefully.

---

6. Capacity Transition

As more associations are added:

- vectors become less independent,
- interference increases,
- margins shrink,
- and correction weakens.

Eventually the system reaches capacity.

At Capacity

At capacity:

- there are just enough degrees of freedom to satisfy the required associations,
- nearly every available direction becomes important.

The interference structure becomes delicate.

Small changes:

- strongly affect solutions,
- alter retrieval geometry,
- and amplify noise.

SGD at Capacity

SGD now has fewer “easy” directions remaining.

Updates increasingly:

- compete,
- undo previous adjustments,
- and redistribute interference.

Training becomes slower and more unstable.

Moore–Penrose at Capacity

The pseudoinverse solution also becomes unstable because:

- small singular directions are amplified,
- tiny geometric inconsistencies matter.

At capacity:

- SGD and Moore–Penrose often behave similarly,
- because there is little remaining freedom.

---

7. Over-Capacity Behavior

Now suppose:

- there are more associations than stable independent directions.

The system becomes overloaded.

SGD Over Capacity

SGD can no longer satisfy all examples simultaneously.

Each new example:

- pulls the weight vector slightly,
- altering previous fits.

The resulting memory geometry depends heavily on:

- learning rate,
- example order,
- batch structure,
- recency,
- and optimization dynamics.

The last few examples shown can significantly bias the solution.

The weight vector becomes a compromise interference field.

Moore–Penrose Over Capacity

Moore–Penrose again becomes advantageous because:

- it globally balances all constraints,
- distributes error more evenly,
- and minimizes unnecessary norm growth.

Instead of chasing recent gradients,
it computes:

- the least-energy compromise solution.

---

8. Error Correction Over Capacity

Error correction changes dramatically.

Under capacity:

- many correct projections cooperate.

Over capacity:

- too many associations compete for the same directions.

The system develops:

- crosstalk,
- ambiguous retrieval,
- unstable margins,
- and anisotropic errors.

Some perturbations:

- are still corrected well.

Others:

- strongly amplify.

Error correction becomes directional.

---

9. Systems of Weighted Sums

A neural network layer computes:

$$y = Wx$$

This is not merely:

- many separate neurons.

It is a coupled interference system.

---

10. Projection Fields

Expand by columns:

$$Wx = \sum_i x_i w_i$$

where w_i is column i of W.

This reveals something important.

Each neuron:

- emits a projection pattern into the next layer,
- scaled by its activation intensity.

The next layer accumulates all emitted patterns.

The output layer state is therefore an interference pattern.

---

11. Under-Capacity Interference Fields

Under capacity:

- many neurons support the same emerging pattern,
- interference is cooperative,
- retrieval basins are broad,
- and noise correction is strong.

The network has redundancy.

Multiple partial projections:

- reinforce each other,
- stabilizing computation.

---

12. At Capacity Interference Fields

At capacity:

- there are just enough patterns to produce the required interference geometry.

The system becomes fragile.

Small perturbations:

- disrupt coherent reconstruction,
- alter cancellations,
- and reduce robustness.

The network loses redundancy.

---

13. Over-Capacity Interference Fields

Over capacity:

- too many desired associations compete,
- correct interference patterns cannot all coexist cleanly,
- and projection fields begin conflicting.

The network compensates through:

- compromise geometries,
- distributed superposition,
- sparse activation,
- and low-rank organization.

But interference becomes increasingly messy.

---

14. Singular Directions and Dominant Modes

Real systems do not use all directions equally.

Certain projection patterns become dominant.

These correspond to:

- strong singular vectors,
- recurring features,

- stable interference modes.

Training naturally amplifies:

- coherent,
- reusable,
- high-energy directions.

This is one reason large networks can remain functional even when massively overparameterized.

---

15. Nonlinearity as Interference Routing

Nonlinearities such as ReLU do more than add nonlinearity.

They selectively gate:

- which projection fields continue propagating,
- which interference modes survive,
- and which patterns are suppressed.

Deep networks therefore repeatedly perform:

1. projection,
2. interference,
3. selection,
4. recombination.

---

16. Practical Interpretation

A weighted sum is best viewed not as:

- “adding weighted numbers.”

Instead it is:

- projecting a learned pattern,
- measuring similarity,
- accumulating interference,

- suppressing orthogonal structure,
- and storing distributed associations.

A weight matrix is then:

- a coupled system of interacting projection fields.

Dense neural computation emerges from:

- constructive interference,
- destructive interference,
- distributed memory superposition,
- and adaptive spectral geometry.

From this viewpoint, neural networks resemble:

- programmable interference systems,
- learned harmonic decompositions,
- and distributed associative memories

far more than collections of isolated scalar neurons.